****

**EPAM Training Center**

**DWH**

**Tasks 3**

**Report**

**Minsk, 2017**

Revision history

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 02.11.2017 | 0.1 | Description of workflow (Tasks 3) | Olga Hilko |
|  |  |  |  |

Contests

[2 Introduction 4](#_Toc497407519)

[2.1 Purpose 4](#_Toc497407520)

[2.2 Prerequirements 4](#_Toc497407521)

[3 Heaps 5](#_Toc497407522)

[3.1 Tables 5](#_Toc497407523)

[3.2 Segments 7](#_Toc497407524)

[4 IOT 8](#_Toc497407525)

[5 ICT 10](#_Toc497407526)

[6 Hash clustered tables 12](#_Toc497407527)

[7 Row migration 14](#_Toc497407528)

[7.1 Migration 14](#_Toc497407529)

[7.2 Channing 14](#_Toc497407530)

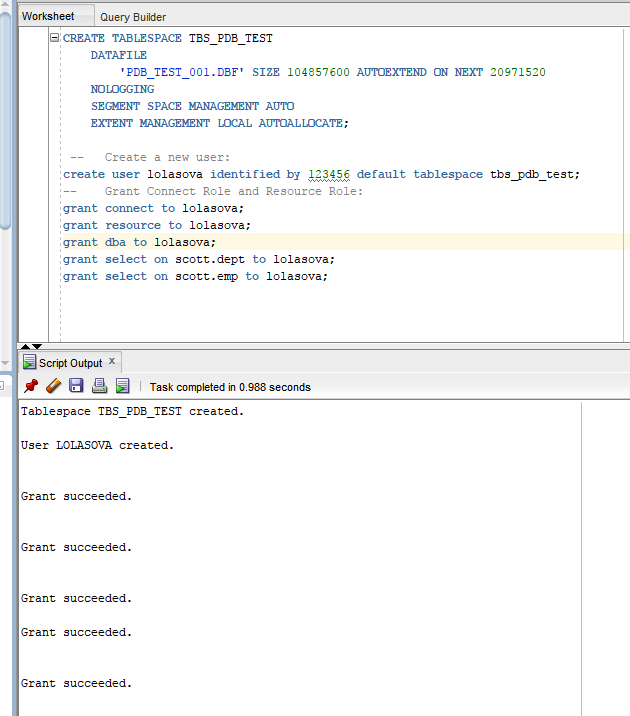
# Introduction

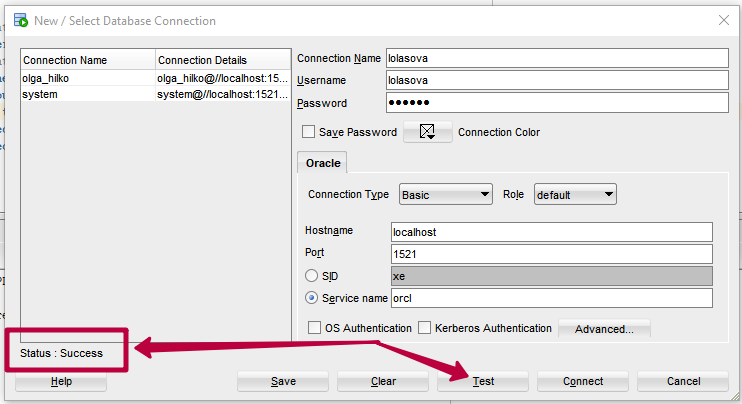
## Purpose

This document includes the results of completed task 3.

## Prerequirements

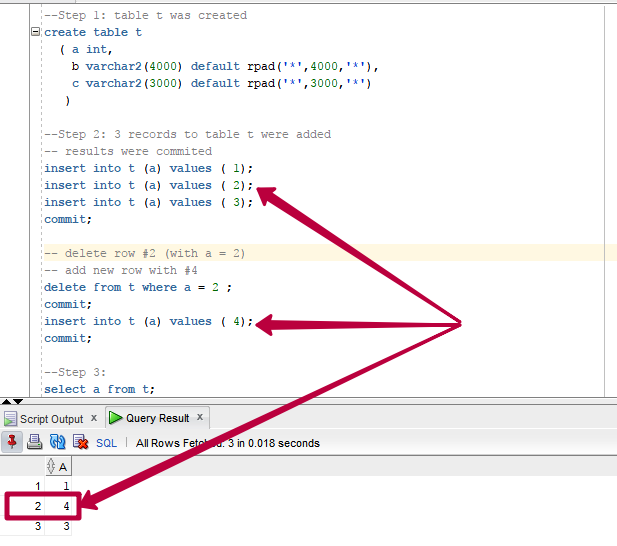
New tablespace and user were created.



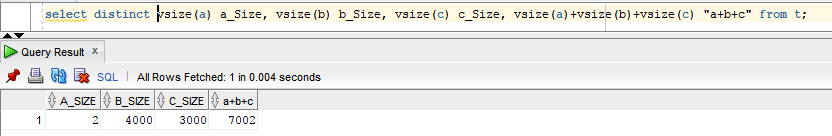


# Heaps

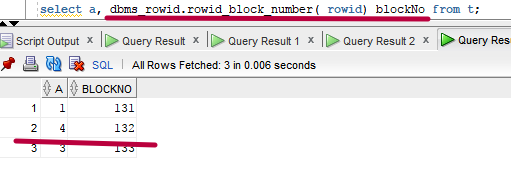
## Tables



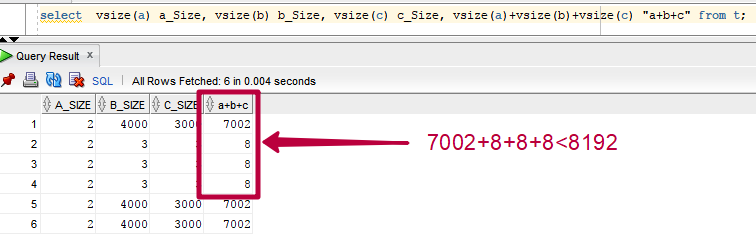
Result: new row has replaced the deleted row. We have db block size (8192 byte) less then our row length (7002 byte). One row occupies 1 block.



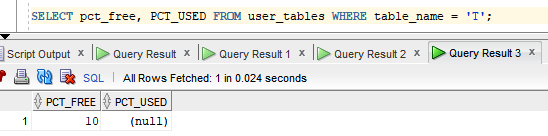


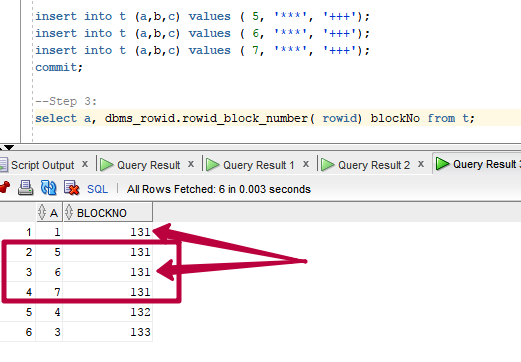


Let’s paste 3 new rows with less size (by 8 byte for each one).

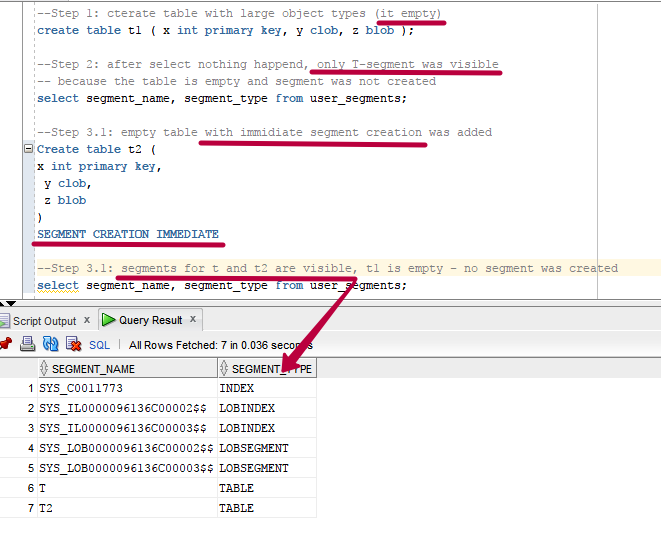


We can see that all of them are located to the first block, because we have free space there. PCT\_FREE coefficient is 10%, that is why there is enough space to paste all new rows.





## Segments



**Result:** Segments were created for non-empty table T2 .

When you create a LOB, you will always get a lob index created (to find the chunks for the lob fast) and a segment that holds the lob data (chunks).

Get info about T2 by command

SELECT DBMS\_METADATA.GET\_DDL('TABLE','T2') FROM dual;

**Result:**

CREATE TABLE "LOLASOVA"."T2"

( "X" NUMBER(\*,0),

"Y" CLOB,

"Z" BLOB,

PRIMARY KEY ("X")

USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 NOLOGGING

STORAGE (INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645

PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1

BUFFER\_POOL DEFAULT FLASH\_CACHE DEFAULT CELL\_FLASH\_CACHE DEFAULT)

TABLESPACE "TBS\_PDB\_TEST" ENABLE

)

SEGMENT CREATION IMMEDIATE

PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255

NOCOMPRESS NOLOGGING

STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645

PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1

BUFFER\_POOL DEFAULT FLASH\_CACHE DEFAULT CELL\_FLASH\_CACHE DEFAULT)

TABLESPACE "TBS\_PDB\_TEST"

LOB ("Y") STORE AS SECUREFILE (

TABLESPACE "TBS\_PDB\_TEST" ENABLE STORAGE IN ROW CHUNK 8192

NOCACHE NOLOGGING NOCOMPRESS KEEP\_DUPLICATES

STORAGE(INITIAL 106496 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645

PCTINCREASE 0

BUFFER\_POOL DEFAULT FLASH\_CACHE DEFAULT CELL\_FLASH\_CACHE DEFAULT))

LOB ("Z") STORE AS SECUREFILE (

TABLESPACE "TBS\_PDB\_TEST" ENABLE STORAGE IN ROW CHUNK 8192

NOCACHE NOLOGGING NOCOMPRESS KEEP\_DUPLICATES

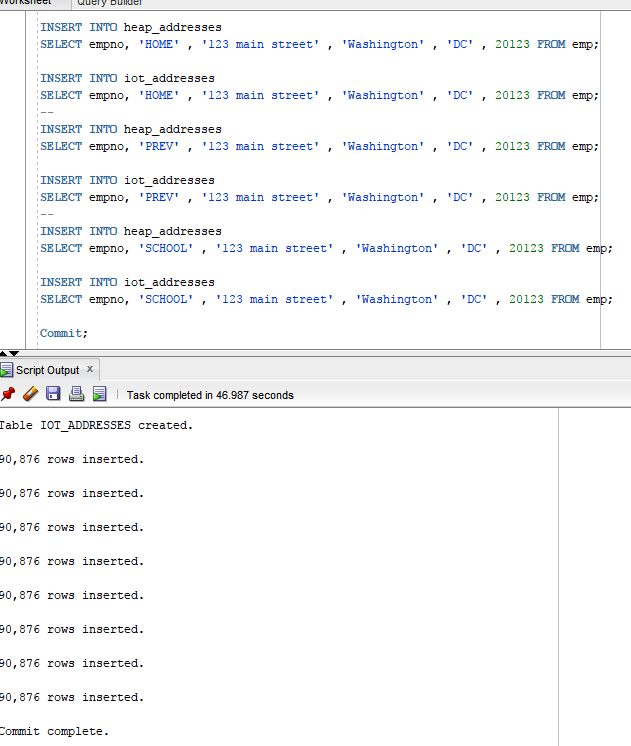
STORAGE(INITIAL 106496 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645

PCTINCREASE 0

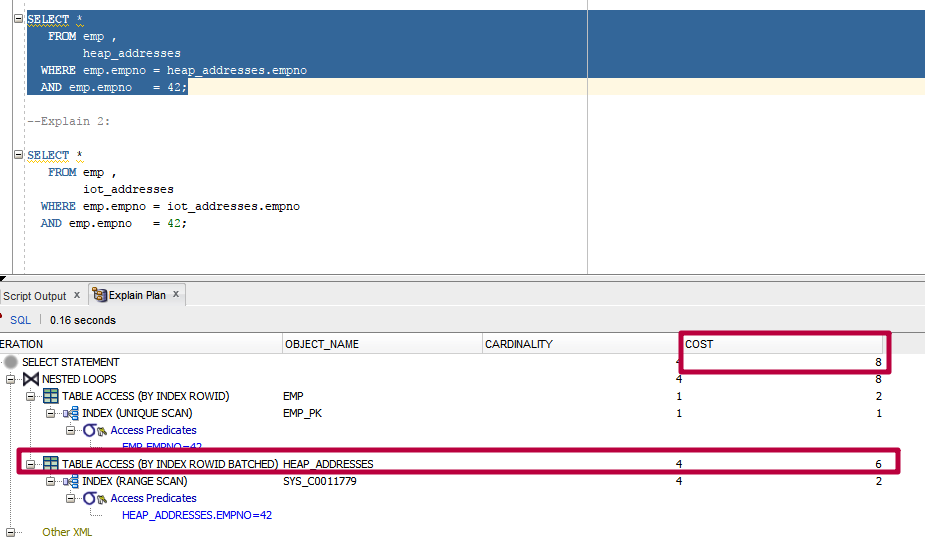
BUFFER\_POOL DEFAULT FLASH\_CACHE DEFAULT CELL\_FLASH\_CACHE DEFAULT)) "

# IOT

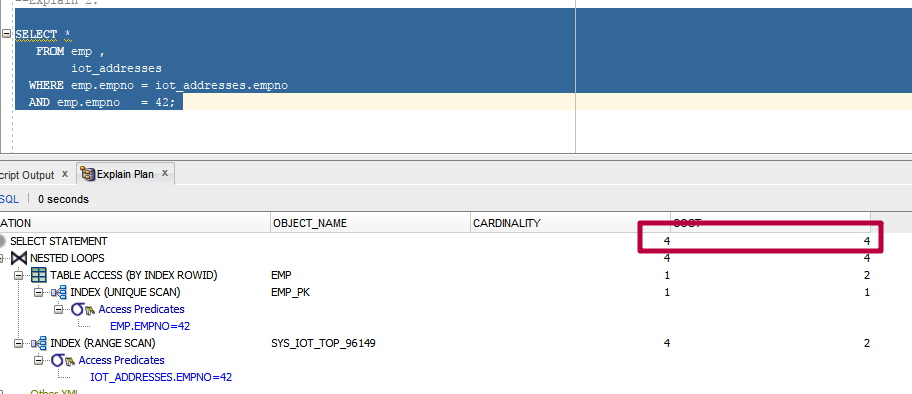
We have created two tables IOT and heap with index.



Execution plan with heap + index table:



Exec plan for IOT



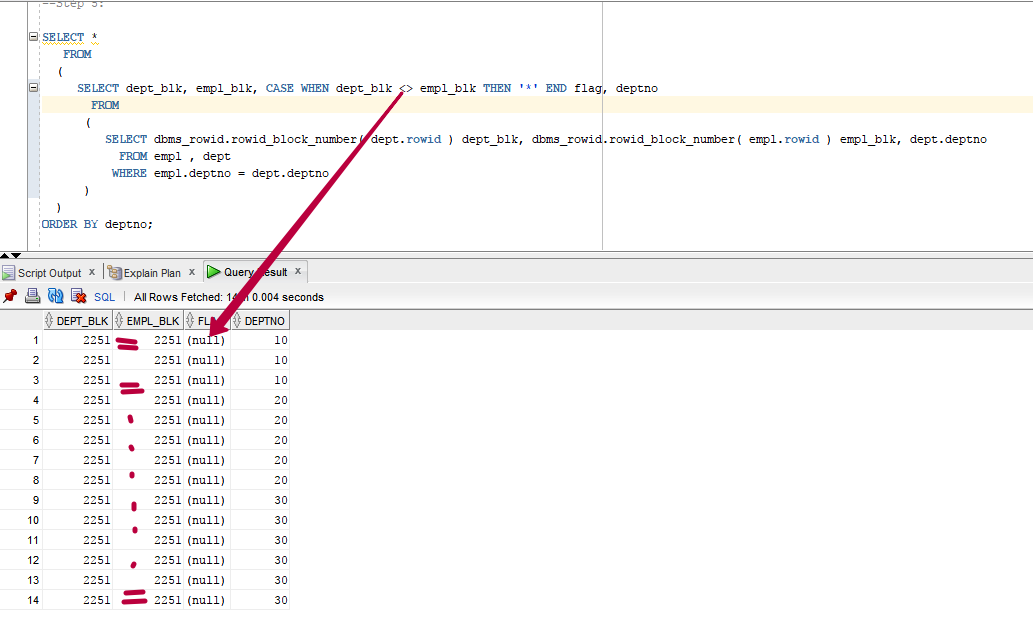
Either heap with index table or IOT has index range scan to be later joined with the main table. But IOT has all data in index (the index is a table itself) and there is no need to have the table access.

# ICT

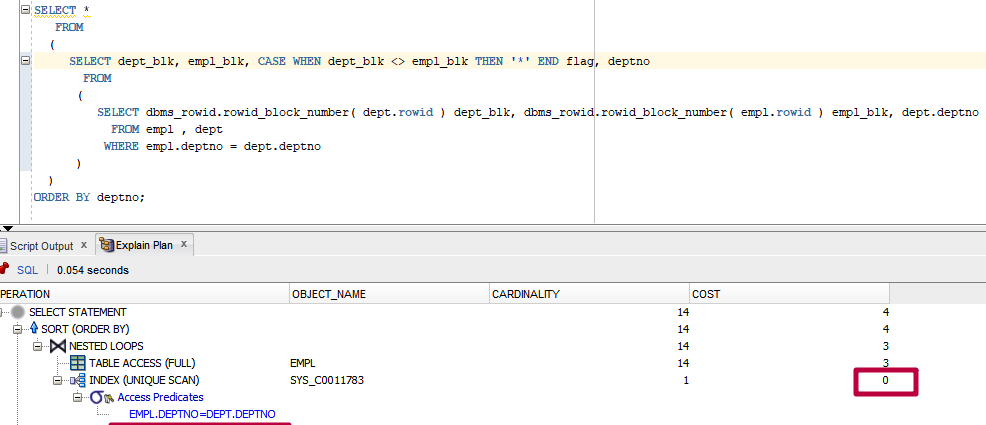
Create cluster with index and two tables with included cluster.



In fact we have one cluster (column), which is shared between two tables. Physical this is one column, which is located in the same one block (there is enough space in the block to accommodate all rows of the cluster). That is why two columns of the select return the same values of block numbers.



Linked data from both of tables are stored behind each other. That in why very little time is wasted for joining them (block is little, data are behind).

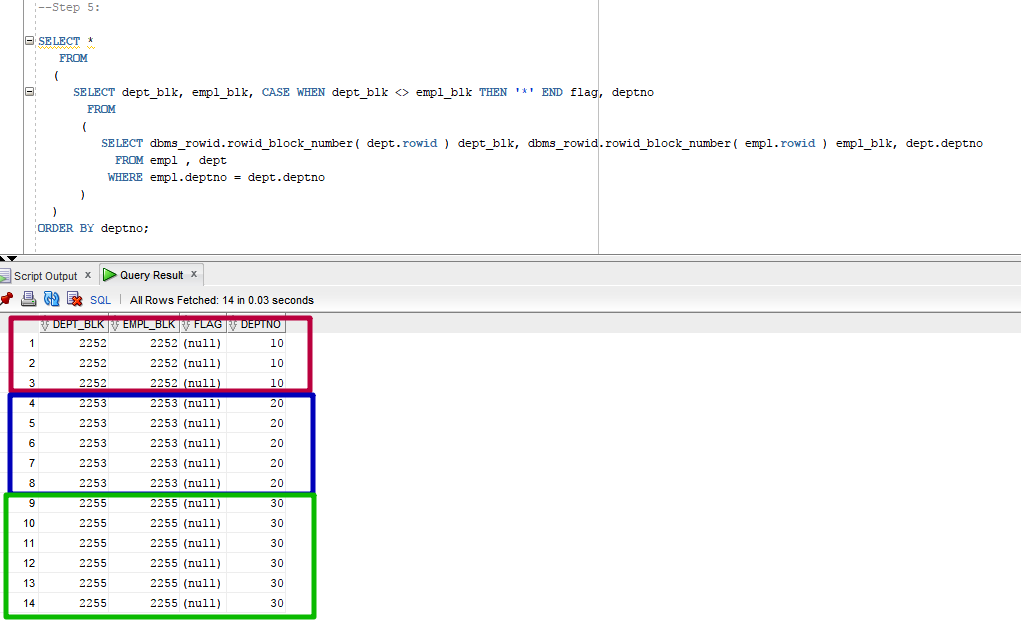


# Hash clustered tables

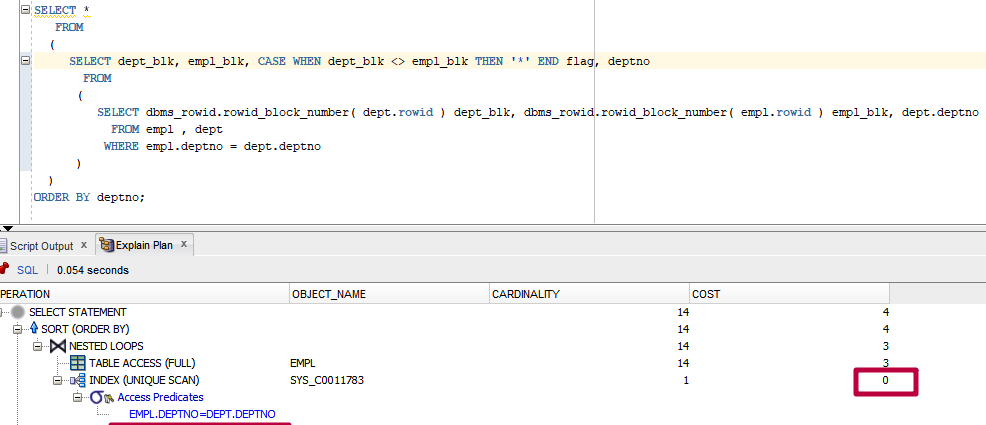
Create hash cluster (no index is needed) and add it two both of the tables. The size is the same as we used in the previous cluster.



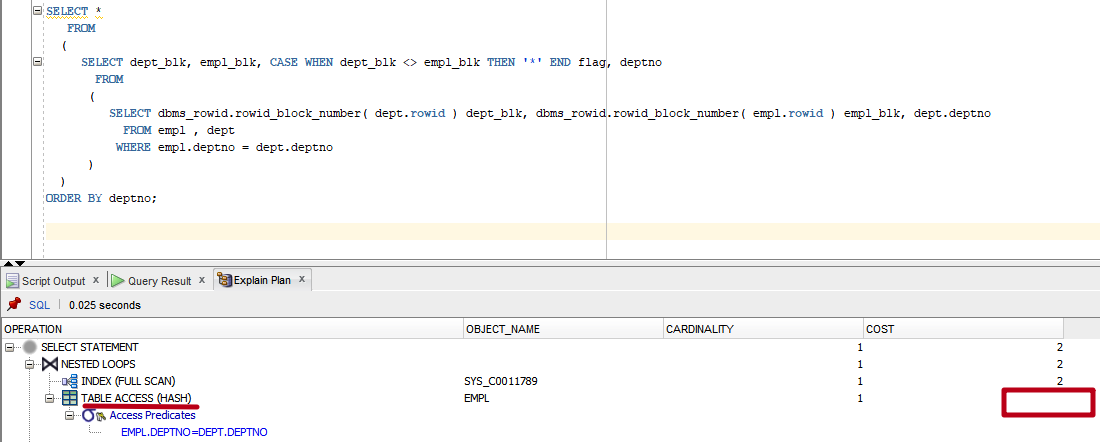
Every id has its own hash key which was generated with hash function. This hash key is stored in the separate block.



Execution plan from the previous task



Current execution plan



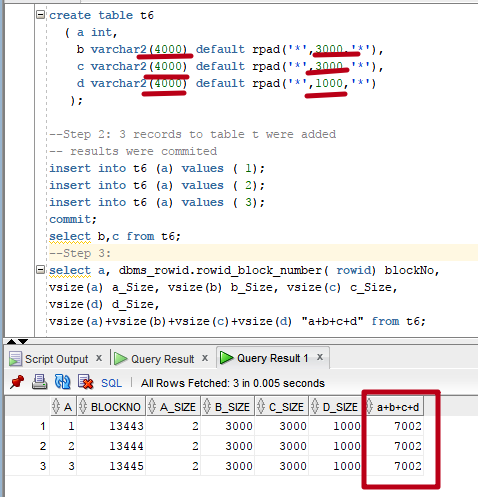
A hash cluster stores related data rows together, rows with the same depno (they have the same hash key value) are stored in the same block. No need to join at all. Data are together.

# Row migration

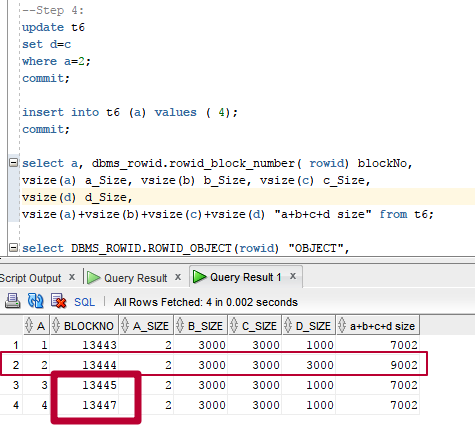
Our block size is 8192 bytes. Reserved space – 10%.

## Migration

Initial state



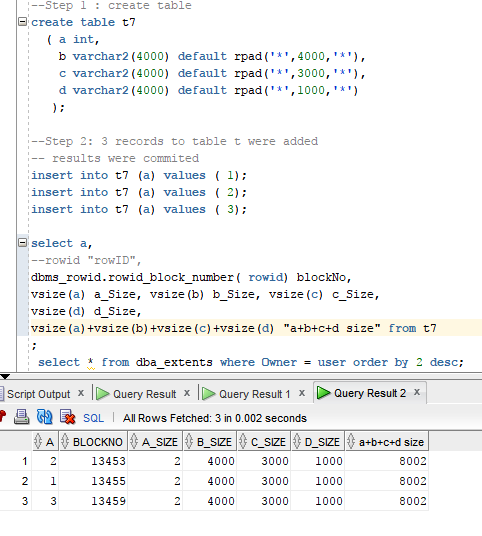
After update row#2 and insert row# 4. Block # 13446 is not shown. It is used by the row #2 (it occupies two blocks). The next row in pasted to the block # 13447.



## Channing

Create table with rows, which are bigger then 0.9\*(block size).

As shown, block numbers are not incremented. Some numbers are missed.



This table occupies two segments.

